1. Introduction

2. Software Utilities

2.1 Interprocess Communication

The authors of this section are Laura Paterno and Stu Fuess. The interprocess communications package chosen will be used by software projects in WBS #s 1.5.3, 1.5.4, 1.5.5, 1.5.6.1 - 1.5.6.10, 1.5.7, 1.5.8, 1.5.9 and 1.5.10

2.1.1 Introduction

Many processes used during Run I need to communicate with each other across the Online Host network. The communication layer was provided by the InterProcess Communications (ITC) package. ITC was written in Pascal and used the Digital DECNet protocol to allow processes to communicate with each other across a local network. A later product, known as the Client/Server Package (CLSPKG), extended the functionality of ITC to provide many of the features necessary for multiple client processes communicating with a single or multiple servers. However, it was only used by a small handful of processes.

2.1.2 Requirements

- Platform independence

Work on VxWorks, UNIX (SGI and DEC Alpha), and NT at minimum.

- Support message-based communication and buffer management

- Designed with OO methodology

For flexibility, extensibility and maintainability

- Written in C++
- Threads support

Have POSIX and NT thread support at minimum

2.1.3 Details

The design of a complex system like interprocess communication with all the necessary requirements listed above is not a trivial one. Therefore, we will be using a product called the ADAPTIVE Communication Environment (ACE)[1] written by Douglas Schmidt, a professor at Washington University in Computer Science.
ACE is a freely available object-oriented toolkit that provides common network programming tasks across a wide range of operating systems. These tasks include interprocess communication, thread and synchronization support, event demultiplexing (receives signals-based, timer-based, I/O-based, synchronization based events) and event dispatching, and other

higher level tasks. It also provides many examples on using the components of ACE.

ACE has many of the features that we want but not all. It provides stream-based and not message-based communication. We would definitely like message-based communication and will have to write that feature ourselves. Also ACE does not handle buffer management and we will need to write this as well. ACE does not have enough examples for TCP/IP sockets, which is how we will be communicating (for the most part) between processes. Therefore, some examples with TCP/IP need to be written. Some documentation is provided in the form of papers published by the author of ACE but this is not sufficient. We will have to provide our own documentation for people to understand.

2.1.4 Schedule

Project	**Duration (weeks)**	**Target Date**
Install ACE		
SGI (Irix 5.3)	2	1 December 1997
SGI (Irix 6.2)	1	9 January 1998
NT (4.0)	1	1 December 1997
VxWorks (5.2)	1	15 January 1998
DEC Alpha Server	1 – 2	15 February 1998
Example TCP/IP Code	4	15 January 1998
Add message-based communication	8 – 12	1 March – 1 April 1998
Add in buffer management	4 – 6	1 April – 1 May 1998
Documentation	4 – 8	1 March – 1 April 1998
Total	26 - 37	

Table 2-1 - Schedule for IPC Package

2.1.5 Resources

1 FTE (100% of time) will be required to complete this project. The workload will be reduced after short period of time (4 – 5 months) to maintenance and occasional enhancements (upgrades) of the product.

2.2 User Interfaces

Databases

The authors of this section are Stan Krzywdzinski and Laura Paterno. All of the online databases will be used in conjunction with most of the online software defined by WBS #s 1.5.3 to 1.5.10.

Introduction

During DØ Run I, two Database Management Systems (DBMSs) were used to manage pertinent information for the online system. They include DBL3 and DEC Rdb.

DBL3 is a CERN database product that provides keyed access to information using the ZEBRA RZ file structure. It was used for managing detector calibration and monitoring information that was stored on-line, compressed and distributed to the off-line processing cluster. This product was chosen in part because of its platform independence.

DEC Rdb[2], a commercial relational DB product from Digital Equipment Corporation, was used for the Tape Log DB, and the Hardware DB. The Tape Log DB kept track of all run, trigger, filter and raw file and tape information taken online. The Hardware DB was used in the online system to provide device access information to the front end system for control and monitoring purpose.

Needs

A wide variety of information will need to be maintained during Run II. Calibration, monitoring, luminosity, run summary, trigger, filter and event information will need to be stored when data taking resumes. In addition, geometry and alignment information will also need to be stored. Access information for hardware devices, used for control and monitoring purposes, will also be needed.

Calibration databases

The Run I DBL3 calibration databases stored information for the Calorimeter, Central Tracking System, Muon System, Level Ø, etc. The largest of them was the Calorimeter database which consisted of 30 (each 300Kbyte uncompressed or 125Kbytes compressed) files. The rest were much smaller in comparison and several were barely used. They were accessed once per run for reconstruction purposes. They were all keyed by run number with a secondary key by time.

In Run II the Calorimeter will remain unchanged (except for the electronics) and the rest of the systems will either be upgraded or replaced. The Calorimeter calibration will not change but the estimated size of the Run II database is 10 times what it was for Run I. The requirements for the other databases are currently unknown.

Monitoring databases

During Run I many of the monitoring programs also stored information into DBL3 databases. These databases include the DBMON, HV, Luminosity and Run Summary databases. All of these databases will again be needed for Run II.

Controls Database (formerly Hardware DB)

In Run I the Hardware DataBase (HdB) was a single DEC RDB database (18.4 Mb). It was used in the monitoring and control of the DØ experiment. It stored all of the essential attributes of the hardware and provided CDAQ processes with the access path information required to read from or write to a device. It contained information on detector systems, low and high voltage power supplies, cryogenics and argon purity monitoring, and environmental conditioning. The database also served as the master copy of the local databases, which resided in the front-end computers. Fast access and reliability was required for this database. The kind of information stored in the Hardware Database will be needed for Run II.

Tape Log DB

During Run I this database was a multifile DEC RDB database (446 Mb) used to store raw file, trigger, filter and begin-end run information. Information was written into it by several online processes and read by offline processes. Some form of this database will be required for Run II.

Requirements

Below is the list of requirements necessary for all the online databases that will be necessary for Run II.

Be commercially supported.

Allow automatic replication (copying) of a database across DØ supported platforms.

Meet performance requirement needs for Run II

On-line calibration data must be accessible no later than 5 minutes after a calibration run has finished.

Robust and corruption free performance

Others as we are told what they are.

Performance must scale for the expected number of users and size of the database. It should take into account that the database may be replicated to help improve performance.

Work in a fully-distributed, multi-platform environment

Be standards compliant

Must have sufficient Database administration tools to do optimization and setup of the database.

Provide database monitoring capabilities.

Preferences

Provide query access via the Web.

Currently, we are looking at using Oracle as the Run II database. However, this may or may not be the final solution.

Details

The Controls Database will be the first database to be designed and implemented. This also entails writing an interface(s) capable of generating extracted forms of the Controls Database (e.g. for EPICS and other online control and monitoring tasks). The database will need to keep track of the same device specific information as in Run I. Since the database needs to be redesigned new information will probably be added.

Schedule

Once the DEC Alpha system is available in January 1998, development of the online databases can begin. In the meantime, work is going on to understand how to use Oracle. Databases and access/updating code will need to be written for all the databases required for Run II.

Project	Duration (weeks)	Target Date
Purchase Database Software	6	15 February 1997
Install Software	2	1 March 1998
Total		

Table 2-2 - Schedule for Databases

Resources

At least 3 core database developers/managers (1/2 FTE each) will be needed to implement and modify the databases required. In addition, 1 developer per database will be needed to write the software to update and access the database.

The 3 core developers would provide the software for updating and accessing the Controls Database. It will also need to have the information stored into the database as well. This will require that the various detector groups provide someone to fill the database with the information necessary for their detector.

For calibration databases they would be provided by various universities, most likely graduate students, working on that part of the detector.

An additional person per database to maintain and check the integrity of each database during its lifetime is also recommended.

Costs

All database software to be purchased falls under WBS #1.5.1.5 (Online Software Budget). The current plan is to use the Oracle database software from Oracle, Inc running on a DEC AlphaServer. As yet no software has been purchased.

3. Run and Configuration Control

3.1 COOR

3.2 Run Control

4. Event Data Path

4.1 Level 3 Interface

4.2 Data Logger

The authors of this section are Gene Oleynik and Fritz Bartlett. The associated WBS section is 15.5.5

4.2.1 Introduction

This section covers the data logging online system from receipt of events from level 3 processors to submission to the Reconstruction Input Pipeline (RIP) for central archival. The logger is responsible for taking event data from L3, buffering and packaging it for archival by RIP, and assuring all event data for a run is archived to RIP at the end of run.

The main functions of the logger are:

- Receipt of event data from the L3 processors
- Sorting by trigger bits into non-overlapping Physics Streams
- Spooling stream data to disk files and maintaining the file space
- De-spooling completed stream files into the RIP system
- Buffering against RIP downtime and other error recovery
- Providing redundant storage for smaller streams
- Participating in run control to flush the data at end of run
- Providing statistics for monitoring

In this section, “streams” has a specific meaning. Streams divide events up into exclusive classes based on their anticipated physics analysis access. Data from a given stream will be physically stored in Hierarchical Storage Module (HSM) by RIP to optimize access by “freight-train” analysis. Current investigations indicate that there will be on the order of 20-30 exclusive streams that have a logarithmic distribution in events/year.

What the logger will ***not*** do is

- Log to tape locally

- Perform any manipulation of streams in order to optimize RIP archival. This implies that RIP is responsible for optimizing the file streams sent to it to prevent tape drive thrashing, etc.
- Perform any event data manipulation. In particular any byte swapping or other formatting required prior to archival will be performed in level-3 upstream of the logger.

The logger will runon multiple computers at the experiment to accommodate the required aggregate 26 MB/s bandwidth and to provide fault tolerance. The event data is spooled to disk files on these computers, and completed files are shipped to central robotic storage across optical fiber links via RIP.

The logger maps event trigger type bits to physics streams and writes the event data to the appropriate stream file. There will be one stream file open for each active stream during a run. When a file reaches a maximum partition size, it will be submitted to the RIP system (by a RIP defined API). When detecting an end of a Run (expected to be from an end-run “event”), all stream files are closed and immediately submitted to RIP. This implies that RIP must handle file sizes from several Megabytes up to the maximum file partition size. The file submission protocol between RIP and the logger includes an acknowledgment from RIP when the file has been successfully archived to tape. The local copy of the file is not deleted until the acknowledgment has been received.

It is important that some of the smaller streams be redundantly archived since their loss, such as due to a damaged tape, would have a great impact on the physics analysis they represent. The logger will need to duplicate these files to a redundant store, perhaps a raid disk array.

The logger must buffer against RIP downtime for up to six hours. This requires several Gigabytes of storage. For three data logging machines this would be on the order of 100 GB/machine (1-3 1999 disks/machine). The logger will probably need to respond to run-control commands, for example a manual close/flush. The logger will be configured at startup through COOR messages.

Incoming data will need to be buffered in the local memory. It is imagined that the Computing Division Data Flow Manager (DFM) buffer management/service provider software will be used to provide the memory management.

The logger will be the source of sample events for online monitoring. It will queue event data through DFM to the Hoist server software, which will serve events over the network. This DART software was used during the last Fixed Target running period, and provided non-blocking event data sampling, and, so, will not impact mainline logging.

4.2.2 Requirements

The data logging system must satisfy the following requirements:

- Keep up with the continuous incoming and outgoing data rates of 13 MB/s (26 MB/s aggregate)
- Buffer against RIP downtime and assure all received event data is archived
- Provide for redundant archiving of the smaller streams
- Run on the platform of choice for logging

4.2.3 Dependencies

The following separate software is required for completion of the logger:

- L3 event input interface – required 2^{nd} quarter '98
- Event trigger type format – 2^{nd} quarter '98
- RIP interface – 3^{rd} quarter '98
- Interface to run-control/COOR – 2^{nd} quarter '98
- Buffer management interface – 2^{nd} quarter '98
- DA Monitoring interface (for statistics) – 3^{rd} quarter '98
- Hoist interface - 3^{rd} quarter '98

4.2.4 Schedule

The project has the following milestones:

Choose platform Delivered Platforms	1^{st} Quarter '98 2^{nd} Quarter '98
Software	3^{rd} quarter '98
Purchase spooling and redundant disk drives	1^{st} Quarter '99

Figure 1 - Milestones

4.2.5 Resources/Manpower

The Computing Division will provide the manpower for this effort. D0 will provide a consulting contact to work with the Computing Division. This project is projected to take 5-6 person months for design, coding and testing.

The Computing Division has developed logging software for the last Fixed Target program that has functionality similar to that required by D0. In particular the DART software DOT and DUF provided event spooling to disk and despooling to tapes with routing based on trigger type. If not the software, some of the concepts and expertise from this software can be reused.

4.2.6 Costs

Costs are for disk drives for buffering and covering RIP down time and redundant storage of small streams and up to three logging machines.

- Glitch, spooling, and raid redundancy disk $50K (wild guess)
- Logging machines $50

4.3 RIP Interface

4.4 System Monitoring

4.5 Event Distribution

4.6 Event Monitoring

4.6.1 Framework

4.6.2 Detector

4.6.3 Physics and Expressline Analysis

5. Calibration

5.1 Framework

Section Author: Iain A. Bertram.

WBS Number 1.5.9.

Tuesday, 16 December 1997

The Calibration Framework is a software package that will run control the various detector calibration tasks. It will be required to initiate each of the sub-detectors calibration tasks, return the results, and notify the user of any errors that occurred during the calibration task. In addition the framework will be required to compare the results with previous calibration runs, provides standard plots of the results, and store the results in the online calibration database. In some cases the detector calibration tasks will adjust the parameters that will be downloaded to the detector at the beginning of each run, if this occurs the Framework will be required to adjust all relevant database entries and down-load files (WBS 1.5.4).

5.1.1 Requirements

The Calibration Framework will be required to satisfy the following:

- Will run on the online DEC alpha system.
- Will use standard GUI interface and set of plotting tools that work on all supported platforms in the control room (PC's and Workstations), possibly developed using the Python language.
- Database tools to enter information into the database (written in Python or C++) see Section????.
- Database tools to review information already stored in the database. These tools will need to be flexible enough to allow experts to use the calibration information for debugging detector problems.
- Disk space for the Online Calibration database (see next section).

In addition the following software resources will be required for the framework:

- Python GUI toolbox for designing user interface
- Database tool box for implementing database access routines

5.1.2 Schedule

- April 1998: First Version of the Code interfacing the Silicon Calibration Task with the Database.
- Undetermined

5.1.3 Resources

- Physicist: 0.2 FTE 1 year

5.1.4 Dependencies

The Calibration Framework will require the following:

- A working Online Database.
- Operational detector calibration tasks (See Section **Error! Reference source not found.**)

detector before they can be completed.

5.2 Detector Components

Each of DØ's sub-detectors will require a calibration tool that measures the necessary calibration constants for that detector. The tool will be required to issue the commands required to run the hardware required by the calibration task, collect the data, analyze the data, provide the resulting calibration constants and report any errors in the calibration.

The requirements of the detector calibration tools will vary depending on the use that the calibration information will be used for.

The sub-detectors that will require calibration tools are: the calorimeter, the silicon tracker, the central fiber tracker, the muon system, the pre-shower detectors (central and forward), the forward proton detector, and the luminosity monitors (unsure if this detector plans on having calibration runs).

The calorimeter's calibration tool will be based on the Run I tool which already exists. This tool needs to be updated for the new calorimeter electronics and software standards that will be used in Run II. All other detector calibration tools need to be written.

5.2.1 Requirements

The requirements depend on the detector being calibrated and in most cases are yet to be determined.

- Calorimeter
  - ♦ The task is required to be completed within 5 minutes.
  - ♦ The tool will measure both pedestals and gains. The measurements will be carried out on alternate days during running.

- ♦ The tool will measure the mean, the RMS width, and ten monitoring histograms and report an error flag for each calibration run.
- ♦ Using the data collected during every pedestal runs the tool will calculate a mean and zero-suppression limit for each channel to be downloaded to the calorimeter.
- ♦ 0.8 Gigabytes of disk space to store the calibration constants in the online database.
- Silicon
- ♦ The tool will measure both pedestals and gains. The measurements will be carried out on alternate days during running.
- ♦ The tool will measure the mean, the RMS width, and ten monitoring histograms and report an error flag for each calibration run.
- ♦ 2.4 Gigabytes of disk space to store the calibration constants in the online database.
- Central Fiber Tracker and Preshower Detectors
- ♦ To be determined. The tools will probably be similar to the Silicon tools, which are based on the SVXII readout chip.
- Muon, Forward Proton Detector, and Luminosity Monitoring
- ♦ To be determined.

5.2.2 Schedule

- The test version of the silicon calibration tool will be required for testing of the calibration framework by April 1998.
- Undetermined

5.2.3 Resources

It is assumed that the various detector systems will be providing the hardware required for calibrating the detector. This TDR is only concerned with the software requirements.

- 0.2 FTE Physicist per detector (7 detectors)

5.2.4 Dependencies

5.3 Database

The Online Calibration Database will be used to store the results of each calibration run for all of the sub-detectors as well as a set of standard calibration constants used to check the performance of the detector. Access to the online database must be fast enough so that is does not provide a significant source of dead time.

A copy of the Online Calibration database will be kept in the Off-line database.

5.3.1 Requirements

Efficient and fast access tools.

~5 Gigabytes of disk space (Guess)

5.3.2 Schedule

A test online database will be required by the beginning of March 1998 so that the development of the Calibration Framework can proceed. Additional schedule requirements are undetermined.

5.3.3 Resources

It will be assumed that the database group will provide the resources required for the calibration database.

5.3.4 Dependencies

detector before they can be completed.

6. Control and Monitoring

6.1 Run I Legacy Code

The author of this section is J. Frederick Bartlett. The WBS section is 1.5.6.1

6.1.1 Introduction

During Run I, the majority of the process variables in the DØ control system were serviced by Motorola 68K and IBM PC front-end processors which were located on a local, dedicated token-ring LAN (*Local Area Network*). These processors executed versions of the ACNET control system software, developed by the Fermilab Accelerator Division, that communicated with host-level processes via the ACNET control protocol. Since the host computers for the DØ control and monitoring system were located on an Ethernet LAN, the DØ Controls Group built DEC microVax-based gateways to handle message traffic between the token-ring and Ethernet LANs.

The API (*Application Program Interface*) for the DØ control and monitoring system was a library of routines collectively referred to as CDAQ (*Control Data AcQuisition*) which provided repetitively scheduled and one-time access to lists of process variables. A process variable name, which was composed of a device/attribute string, was translated to the network path description and ACNET access parameters required by the ACNET message protocol through a relational database.

The majority of the DØ -specific software on the host processors was written in DEC Pascal, which is an extended dialect of the ISO standard Pascal language. The gateway and front-end software was written in either the PSOS[3] or VaxEln[4] dialects of Pascal.

6.1.2 Requirements

Since many of the process variables associated with the Run I detector are carried over to Run II and since the existing front-end processors and the connecting token-ring LAN worked very well in Run I, there is no need to replace these components for Run II. The ACNET sub-net must be integrated into EPICS (Experimental Physics and Industrial Control System), the control system selected for RUN II, in a seamless manner. Access to both ACNET and EPICS process variables, as viewed by application processes running on the host-level computers, must be identical.

6.1.3 Design

Merging the ACNET and EPICS sub-nets can be achieved in either of two ways:

- Gateways–The gateways, in addition to their performing the function of framing messages passing between the Ethernet and token-ring LANs, could also become EPICS channel access servers and transform the message contents between the EPICS and ACNET control protocols. This is feasible since the two protocols possess similar functionality.
- API–The application interface to EPICS could use an optional EPICS product, called CDEV, which inserts an object-oriented layer above the EPICS channel access client. The CDEV layer is specifically designed to manage control sub-nets with different protocols. This is possible because most process control systems, as in the case of EPICS and ACNET, have similar functionality.

6.1.4 Gateway conversion

The Run I gateway program, which is written in VaxEln Pascal, executes under the VaxEln operating system, a discontinued product of the Digital Equipment Corporation, and runs on the discontinued microVax computer. In addition, the gateways communicate with host-level processes via the DECNET network protocol, which is not supported on most non-DEC platforms.

In order to be compatible with contemporary host platforms and in order to minimize maintenance problems during Run II:

- The gateway program must be recoded in either the C or C++ language, for which there are available compilers.
- The microVax processors must be replaced by ones similar to other front-end systems used at DØ.
- The operating system must be changed to VxWorks[5], the same as that used on other EPICS front-end systems.
- The network message protocol must be changed from DECNET[6] to TCP/IP, the standard for most contemporary platforms.

If it is decided that the gateways should also serve as converters between the EPICS and ACNET protocols, this will executed as a later phase and will require the installation of an EPICS channel access server software in the gateways and the design of a protocol converter,.

6.1.5 CDAQ conversion

The Run I application program interface to the control system, CDAQ,:

- Is coded in an extended dialect of ISO Pascal
- Uses the DECNET network message protocol
- Is interfaced to the DEC VMS[7] operating system
- Uses the DEC RDB relational database to store the process variable access parameters.

In order to run on multiple platforms and be maintainable, CDAQ must:

- Be recoded in the C or C++ language
- Use the TCP/IP network message protocol
- Use the POSIX standard operating system interface
- Use a relational database which operates on contemporary host-level processors

In addition, the existing CDAQ process variable database must be converted to a new database format.

6.1.6 CDEV modification

Installing ACNET as a control sub-net under CDEV requires building a set of wrapper routines that convert CDEV's sub-net call interface to the appropriate sequence of CDAQ function calls.

6.1.7 Dependencies

This section is dependent upon the Shea/Goodwin front-end computers (WBS 1.5.6.7).

6.1.8 Schedule

The schedule has the following milestones:

.

Milestone	Date
Gateway - Phase 0	
CDAQ conversion	
CDEV extension	
Gateway - Phase 1	

Table 6-1

6.1.9 Resources/Manpower

The principle resource for this project is programming time according to the following table:

Project	Effort (Person Weeks)
Gateway conversion design - Phase 0	1
Gateway conversion coding - Phase 0	2
Gateway conversion testing - Phase 0	1
Gateway conversion design - Phase 1	2
Gateway conversion coding - Phase 1	4
Gateway conversion testing - Phase 1	1
CDAQ conversion design	1
CDAQ conversion coding	3
CDAQ conversion testing	1
CDAQ database conversion design	1
CDAQ database conversion coding	2
CDAQ database conversion testing	1
CDAQ database loading	1
CDEV modification design	1
CDEV modification coding	2
CDEV modification testing	1
Total	25

Table 6-2

6.1.10 Expenditures

6.1.11 Equipment

The principle equipment cost will be the purchase of three VME-based processor boards for the gateways, at a cost of approximately $5K per processor.

6.1.12 Software

There are no commercial software costs

6.2 High Voltage System

The author of this section is J. Frederick Bartlett. The WBS section is 1.5.6.2

6.2.1 Introduction

The high voltage sub-system, during Run I, was partitioned by the major detector components, i.e. calorimeter, muon, etc. Each high voltage partition consisted of an ACNET front-end processor, a high-voltage master processor, and one to four subordinate processors. Each subordinate processor "supervised" a single VME crate that contained a maximum of six high-voltage modules with each module housing eight Cocroft-Walton voltage generators for a maximum count of forty-eight channels per crate. Several types of voltage generator pods existed, supplying different voltage ranges and polarities.

The subordinate processors directly managed the individual high-voltage hardware channels, which were capable of executing only simple operations. The subordinate processors presented to the master processor a high-voltage channel object which was capable of synchronized ramping, trip recovery, transient current spike detection with optional trip, high resolution history logging, detector-specific management functions, and other features.

The master processor was able to access a restricted block of VME address space, which contained the channel objects, in each of the subordinate crates through a Fermilab-designed vertical interconnect module. This mapping allowed the master processor to access a data structure associated with each channel, to initiate state transitions for each channel, and to trigger the update cycle for the subordinate processors. In addition, channels could be associated in logical groups so that the trip of any member channel could force the trip of all channels in the group. For the vertex detector, where a voltage breakdown between wires could be destructive, the master processor guaranteed that, during the ramping process, specified voltage difference limits were never exceeded.

A global high-voltage status program provided, in a single, high-density display, a status summary of all of the high voltages channels. A second operator program provided an interactive display that could control as well as display detailed status for a group of high-voltage channels. In Run I, each detector component had a copy of the control program.

6.2.2 Requirements

For Run II, only the calorimeter and parts of the muon detector will retain their original high-voltage configuration. In particular, the vertex detector, with its strict channel voltage difference limits, is no longer present. There are several new voltage generator units covering the low voltage ranges required by the silicon vertex and scintillating fiber electronics. The total number of high-voltage channels is not expected to change significantly.

The platform dependence of the present system must be eliminated, which implies a major rewrite of all of the host-level software components. It may also be more economical to move entirely to the use of EPICS as the control layer rather than the ACNET sub-net with its dependence upon the Ethernet/token-ring gateways and the now-outdated PSOS operating system.

6.2.3 Design

6.2.4 Front-End Server

There are two options. The simplest, although not necessarily the most economical over the long term, is to retain the existing, three levels of front-end processors (ACNET, master, and subordinate). The other option is to eliminate the ACNET control front-ends and the master processes entirely and to move the subordinate processes to EPICS front-end nodes. This eliminates two levels of processors in the current hierarchy with concomitant simplification and improved reliability. The existing channel state machines would require recoding in the C or C++ language to execute under the VxWorks[8] operating system and the channel objects would then be incorporated into the EPICS record/device hierarchy.

6.2.5 Global Status Display Program

The global high-voltage status display program for Run II is primarily a re-write of the Run I version to a platform-independent form with the following major alterations:

- Conversion to the C or C++ language with, possibly, a Python top layer
- Conversion to the POSIX standard operating system interface
- Replacement of the CDAQ control interface with EPICS (cf. **Error! Reference source not found.**)
- Use of a multi-threaded computational model
- Replacement of the character-cell graphics with Motif or a similar graphics standard

6.2.6 Operator Control Program

The operator high-voltage control program for Run II is also primarily a re-write of the Run I version to a platform-independent form with the following major alterations:

- Conversion to the C or C++ language with, possibly, a Python top layer
- Conversion to the POSIX standard operating system interface
- Replacement of the CDAQ control interface with EPICS (cf. section **Error! Reference source not found.**)
- Use of a multi-threaded computational model
- Replacement of the character-cell graphics with Motif or a similar graphics standard
- Major alterations to the control algorithms

6.2.7 Dependencies

This section is dependent upon the Legacy Code from Run I (WBS 1.5.6.1) and the Shea/Goodwin front-end computers (WBS 1.5.6.7).

6.2.8 Schedule

The schedule has the following milestones:

Milestone	Date
Server	
Global display	
Operator control	

Table 6-3

6.2.9 Resources/Manpower

The principle resource for this project is programming time according to the following table:

Project	Effort (Person-Weeks)
Server design	2
Server coding	3
Server testing	2
Global display design	1
Global display coding	2
Global display testing	1
Operator control design	4

Operator control coding	4
Operator control testing	2
Total	21

Table 6-4

6.2.10 Expenditures

6.2.11 Equipment

If the current HV front-end servers are replaced with EPICS-based systems, each of the current slave processors must be replaced with a processor having an integrated Ethernet controller at a cost of approximately $5K per processor. Timing studies will determine whether a single processor is capable of controlling and monitoring more than a single HV crate, using the existing vertical interconnect modules from Run I to map several VME crates into a single processor’s address space.

6.2.12 Software

There are no commercial software costs.

6.3 Significant Event/Alarm System

The authors of this section are Stu Fuess and Laura Paterno. The WBS section is 1.5.?.?.

6.3.1 Introduction

The Run I DØ Significant Event System, also known as the Alarm System, consisted of several processes which worked together to keep track of any problem in the DØ hardware or software while the experiment was running. (A picture here would be good if we have one).

6.3.2 Requirements

- Handle sudden bursts of 10000 alarms without losing one or hanging the system it is running on
- Handle between 40 – 60 client connections

6.3.3 Dependencies

This project is dependent on the interprocess communications package, ACE (WBS #1.5.6.10), the interface to the Controls Database (WBS #1.5.6.8), the User Interface decision (WBS #1.5.6.10) and EPICS (WBS #1.5.6.X).

6.3.4 Schedule

The schedule has the following milestones:

Milestone	Date
Server	

Logger	
Watcher	
Display	
Scanner	

Table 6-5

6.3.5 Resources/Manpower

The principle resource for this project is programming time according to the following table:

Project	**Effort (Person-Weeks)**
Server design	4
Server Implementation	12
Server testing	8
Logger design	2
Logger implementation	2
Logger testing	2
Watcher design	1
Watcher implementation	2
Watcher testing	2
Display design	4
Display implementation	24
Display testing	8
Scanner design	2
Scanner implementation	4
Scanner testing	2
Total	79

Table 6-6

6.3.6 Expenditures

6.3.7 Equipment

The principle equipment cost will be ??.

6.3.8 Software

Are there any commercial software purchases involved?

6.4 EPICS Control System

The authors of this section are Jeff McDonald and Harrison Prosper. The WBS section is 1.5.6.?.

6.4.1 Introduction

For online controls and monitoring of the DØ detector during run II, the EPICS (Experimental Physics and Industrial Control System) controls and monitoring software package will be used. EPICS is a software package written by a collaboration of several national laboratories and industries. The primary contributors to EPICS are Los Alamos National Lab, Argonne National Lab and Thomas Jefferson National Lab (formerly know as CEBAF). EPICS provides the software components necessary to provide a robust real-time, multi-platform control and monitoring system and many of the graphical user interfaces that can be used to provide operator interfaces. EPICS supports several field bus architectures, including VME, CAMAC, and GPIB.

6.4.2 Needs

For the DØ detector, control and monitoring of the individual components is critical to insure data reliability and to protect against detector damage. Critical component information must be obtained and, if necessary, provided to the operators so that any actions necessary to maintain data reliability and a functional detector. There are at least three critical needs of the control system:

- OPI (operator interface)
- IOC (input/output controller, e.g. front end processor)
- LAN (local area network) for communication

6.4.3 Operator Interface

The DØ detector is composed of many subsystems and in turn, those subsystems are composed of individual components. Failure of any one of thousands of components must be reported to the detector operators so that actions can be taken to prevent additional damage or data loss. The operator must be provided with an interface that highlights only necessary or critical information while a run is in progress. EPICS provides a number of interfaces that make this exchange of information possible and simple for the operator.

6.4.4 Input/Output of Detector Elements

A control and monitoring system must provide software access to the individual detector components that make up the system. EPICS does this via the use of the VxWorks[9] real-time operating system in a Motorola 68k processor board. The VME chassis provides the bus for the interaction of VxWorks and EPICS with individual detector elements.

In additional, to providing the functionality required to communication to standard detector devices, it is desired that the control system make support of additional devices possible and simple. EPICS provides this support via standard record and device support routines.

6.4.5 Front End/ Host Communication

The front-end processor boards must distribute information back to the host computers where the operators are located. The EPICS control system component, channel access, is the backbone of the communication between these two systems.

6.4.6 Requirements

Below is the list of requirements necessary for the online control and monitoring of the DØ detector necessary for Run II.

- Be supported and widely used.
- Easily extended to new detector elements and detector specific components.
- Meet operator needs for Run II
- ♦ Immediate and specific information about device failure, device out of tolerance, or device error.
- ♦ Robust interface with an “error driven” design.
- ♦ Adaptability.
- ♦ Operator interface works on a variety of platforms.
- Communications standard between front-end computer and hosts.
- Front-end computers use a standard, commercially support real-time kernel.
- Be standards compliant.
- It is desired that any system conform to computing standards outlined by the POSIX standard.

There are many supported record and device types already in EPICS but if these types don’t meet our needs, more specific types must be written

6.4.7 Dependencies

6.4.8 Schedule

The schedule has the following milestones:

Milestone	Date

Table 6-7

6.4.9 Resources/Manpower

The principle resource for this project is programming time according to the following table:

Project	**Effort (Person Weeks)**
Total	0

Table 6-8

.

6.4.10 Expenditures

6.4.11 Equipment

An offline host computer is necessary for cross-platform development and the user interface. Each Motorola 68k processor board costs about $5K. A VxWorks run-time license can be purchased for about $400 per processor.

6.4.12 Software

There are no commercial software costs.

The EPICS system is freely distributed by the EPICS collaboration. DART Secondary Data Acquisition

The author of this section is Stan Krzywdzinski. DART DAQ systems will be implemented and used under the software project defined by WBS #1.5.6.5, which is a sub-project of the Control and Monitoring project.

6.5 DART Secondary Data Acquisition

6.5.1 Introduction

DART (*Data Acquisition Real Time*) was developed on Unix (host) and VxWorks (front end) platforms by the Fermilab Computing Division, in collaboration with several fixed target experiments. DART consists of a number of components (products), which are not constrained to any particular hardware architecture. Components can be selected according to needs, then tailored and extended. Tailoring involves writing script and program interfaces using functions provided by the components.

6.5.2 Requirements

In the period before Run II, DART will be used for testing and debugging the various components of the DØ detector. During the Run II, the stand-alone DART systems (one per sub-detector and independent of the DAQ proper), running in parallel and asynchronously, will be used for diagnosing, checking out, and monitoring the same components of the detector.

Initially implemented on the SGI/IRIX platform, the DART host-level software will eventually be ported to Digital UNIX platforms.

- Portability across DØ supported platforms used in the Control Path. This requires porting all of DART products, which are needed by DARTDØ hosts, to Digital UNIX.
- Adequate rate of data transfer between front end(s) and a host
- Handling of sub-events, from several front ends, across the Ethernet and their concatenation on a host into a single event in a self-describing format.
- System easy to adapt and/or configure for different sub-detectors

6.5.3 Current status

In 1996/97 the DART components needed to service DA activities in NWA Test Beam were put together, the interfaces written (as DARTDØ test package), and the whole system tested and used. The system is capable of acquiring the event data at a rate of up to ~600 kbytes/sec, from a single front end 68K MVME processor running under VxWorks, over the Ethernet link, to an SGI host running under IRIX 5.

The DARTDØ system features:

- GUI panel on the host, which interfaces a user to run control commands
- Logging data to a disk or/and tape(s)
- Monitoring and displaying of the event data
- Monitoring the data acquisition performance and displaying its parameters
- Despooler, which copies to tape the data files that are logged to a disk

6.5.4 Dependencies

This section is not dependent upon other components of the online system.

6.5.5 Schedule

Currently DART runs under IRIX 5. Porting to IRIX 6 by Computing Division is in progress and should be finished in January of 1998. Porting to DEC Alpha under Digital UNIX should be done after that (porting of DFM and DFM_HOIST products will be done for the Event Distribution of DAQ proper, under WBS #1.5.8).

Multiple sources of data (sub-events) should be implemented in the first quarter of 1998. The already identified components to be run on a host, Ethernet Gateway and Event- Builder, will be designed and coded by the Computing Division. The project includes also packing of sub-events at front ends, and unpacking them on host, for efficient use of the Ethernet link.

The implementation of DARTDØ systems, configured for the commissioning and then monitoring of various sub-detectors, should proceed as needed.

The schedule has the following milestones:

Milestone	Date
Porting DART to IRIX 6	02/01/98
DART support of multiple front-end nodes	03/01/98
Porting DART to Digital UNIX	04/01/98
DART detector-specific modifications	?

Table 6-9

6.5.6 Resources/Manpower

One FTE for 2 months is needed from Computing Division to design and code multiple front-ends capability. One FTE for 2 months is needed from Computing Division to port to Digital UNIX all DART products used by DØ. The Computing Division will also provide consulting and help in setting up and extending the capabilities of DARTDØ systems. One person from DØ, understanding DART, is needed for bringing up, upgrades and maintenance of the sub-detector systems.

The principle resource for this project is programming time according to the following table:

Project	Effort (Person Weeks)
Porting DART to IRIX 6	8

DART support of multiple front-end nodes	8
Porting DART to Digital UNIX	8
DART detector-specific modifications	10
Total	34

Table 6-10

6.5.7 Cost

6.5.8 Equipment

There are no equipment costs associated with this project.

6.5.9 Software

There are no commercial software costs.

6.6 Clock System

The authors of this section are Laura Paterno and Fritz Bartlett. The WBS section is 1.5.6.6.

6.6.1 Introduction

The Run I DØ Clock System consists of both hardware and software components. The hardware system is composed of three module types:

- Phase Coherent Clock (PCC)
- Sequencer
- Selector Fanout

The PCC receives timing signals from the accelerator, which are used to lock a local clock generator and provided this clock to the Sequencer. The Sequencer generates 23 programmable timelines, which are sent to the Trigger Framework, Selector Fanouts, and various sub-detectors. The Selector Fanouts also provide timing signals to the Trigger Framework and to various sub-detectors. Two Sequencers were used during Run I: one for the Muon System and another for the other sub-detectorsystems.

The software used to control the Clock System in Run I consisted of client and server components. Multiple clients could talk to a clock server but only one clock server per sequencer could run at any given time. The server only controlled the PCC and Sequencer. The Selector Fanouts were initialized as a part of the begin-run sequence.

6.6.2 Requirements

In Run II, the DØ Clock Hardware System will not change with the possible exception of the functionality of the Sequencer. The software for the system will, however, require modification. The original code was written in an extended dialect of ISO standard Pascal, required the VMS operating system, and used the Run I InterProcess Communication (ITC) package described in the Software Utilities section.

-
-
-
-
-

The clock server and client software must have the following modifications:

- The source code must be translated to the C or C++ language. This implies some functional redesign to effectively use object-oriented methods.

- The operating system interface must be changed to the POSIX standard for platform independence.
- The inter-process communication package must be replaced by the Run II equivalent, ACE.
- A graphical user interface must be added to the client.
- Support for the new Sequencer must be added if it is upgraded.

6.6.3 Dependencies

The Clock System software is dependent on the new Interprocess Communications package, ACE, (WBS 1.5.6), the Controls Database (WBS 1.5.6.8), EPICS (WBS 1.5.6.4), and the Significant Event System (WBS 1.5.6.3). However, the clock code can be developed without access to the database or the Significant Event System.

6.6.4 Schedule

The schedule has the following milestones:

Milestone	Date

Table 6-11

6.6.5 Resources

The principle resource for this project is programming time according to the following table:

Projects	Effort (Person-Weeks)
Server code	10
Client code	10
Interface	6
Total	26

Table 6-12

One FTE is needed to complete this project in ~6 months

6.6.6 Expenditures

6.6.7 Equipment

There are no equipment costs associated with this project.

6.6.8 Software

There are no commercial software costs.

6.7 Control Database

The authors of this section are Stan Krzywdzinski and Laura Paterno. The WBS section is 1.5.6.8.

6.7.1 Introduction

In Run I the corresponding Hardware Database was a relational one, implemented in DEC Rdb. It was used in the monitoring and control of the DØ experiment. It stored all of the essential attributes of hardware devices and provided the Control DAQ (CDAQ) with the access path required to read from or write to a device. In particular, it contained information on detector systems, low and high voltage power supplies, cryogenics and argon purity monitoring, and environmental conditioning. Being the central database it also served as the master copy for down loading the local databases, which resided in the front-end computers. Nominal device readings and their limits stored in the database were used in the Alarm System.

To enter the data into the database, three high level facilities (written in C or FORTRAN) were provided as well as a library of routines (written in PASCAL) to access the stored data by CDAQ and other applications. The low level access routines, used by the high level ones described above, were written either in RDML embedded in C, or SQL embedded in FORTRAN. In addition, the maintenance, monitoring and performance tools of DEC Rdb were used. Fast access and reliability were required for this database and it lived up well to these requirements.

6.7.2 Requirements

For Run II, the Control database will have to be redesigned in ORACLE, the DBMS chosen and supported by Fermilab. The amount of work involved in learning the new DBMS and redesign the database depends on whether a relational or an object-oriented version of ORACLE is used. Storage of data for the existing detector systems and the new ones being built should be optimized, using object-oriented concepts for repetitive, or derivable from a seed device, device structures. The database access facilities, for entering and extracting data as well as interfaces to the monitoring and control systems to be supported by this database, will have to be uniformly rewritten in C or C++.

In particular, the database should be able to provide extractions in text form for the Control DAQ, to be implemented under EPICS (Experimental Physics and Industrial Control Systems) WBS #1.5.6.4. .

The interfaces at the highest level should be implemented in Python GUI (TkInter).

General requirements listed in a section covering on-line Databases, WBS #1.5.11.3, apply here as well.

6.7.3 Schedule

Learning the DBMS selected should start immediately and in parallel with a thorough review of the devices for all detector systems to be stored in the database. Design of the database and coding the interfaces should be done in close consultation with the control and monitoring systems, which will be using the data from the database. Next is the database population, testing the interfaces and performance studies. Population of the database should be made as easy and non-expert as possible, especially for a massive (batch) mode.

The schedule has the following milestones:

Milestone	**Date**

Table 6-13

6.7.4 Resources/Manpower

Two database experts are needed for the database design. One expert could then implement and develop the related code. Once the whole database system is functioning, but not necessarily all data is entered into the database, one database administrator is needed for its maintenance, performance monitoring, and future enhancement. The principle resource for this project is programming time according to the following table:

Project	**Effort (Person Weeks)**
Learn ORACLE and devices	6
Redesign database	4
Create database	2
Create access routines	8
Test access routines	2
Create extraction routines	6
Test extraction routines	2

User Interface	8
Fill database	8
Total	46

Table 6-14

6.7.5 Expenditures

6.7.6 Equipment

There are no equipment costs associated with this project.

6.7.7 Software

There are no commercial software costs since the laboratory has a site-wide license for the Oracle product.

6.8 Cryogenic Controls Interface

This section was written by Laura Paterno. TheWBS section is 1.5.6.9.

6.8.1 Introduction

Information from the cryogenics control for both the Calorimeter and Muon systems was recording in a database periodically (every few minutes) during Run I. This information was need to help reconstruct data offline into physics objects.

6.8.2 Requirements

- A database is necessary for storing the cryogenics information
- Access to the cryogenics system via Ethernet to retrieve the information.

6.8.3 Design

The DØ cryogenics system (see figure below) will have a new hardware configuration for Run II. The system will consist of Programmable Logic Controllers (PLCs) that are connected to the physical hardware to be read out. The PLCs are connected via Ethernet to the Supervisory Control and Data Acquisition (SCADA) PCs that are used to control and monitor the IO devices. One SCADA PC will be used for each subdetector (Calorimeter, Muon, etc.) requiring cryogenics. Each SCADA will also have a database, which is used to store the information for the subdetector it is controlling/monitoring. The databases can be accessed by any other system that uses the DDE/share, DDE, or ODBC protocols. The online host system will have a database that will be able to directly gather information from the SCADA PC databases. This will either be done via a script that will update the online database or a more sophisticated client/server solution.

6.8.4 Dependencies

The software written will be dependent on the database chosen for the online system for Run II.

6.8.5 Schedule

The schedule has the following milestones:

Milestone	Date

Table 6-15

6.8.6 Resources/Manpower

The principle resource for this project is programming time according to the following table:

Project	Effort (Person Weeks)
Design database	4
Create database	1
Write scripting code	4
Total	9

Table 6-16

One FTE is needed to complete this project in ~2-3 months.

6.8.7 Expenditures

6.8.8 Equipment

A front-end, control interface processor may be required at a cost of approximately $5K.

6.8.9 Software

There are no commercial software costs.

6.9 Detector-Specific Support

The author of this section is J. Frederick Bartlett. The WBS section is 1.5.6.10

6.9.1 Introduction

Each detector section (calorimeter, silicon vertex tracker…) has unique and specific online system requirements for calibration, performance monitoring, status display, and fault diagnosis.

6.9.2 Requirements

In order to minimize the online software development efforts of the detector groups, an extensive library of common functions must be identified and assembled into a library that is easily accessible and well documented. The expectation is that most of the modules in the library will be coded in either the C++ or Python[10] languages.

6.9.3 Design

Most of the constituents of the detector support library fall in one of the following categories:

- Display (graphical) tools
- Process variable access
- Inter-process and thread synchronization
- Status logging
- Run control and synchronization
- Frameworks (common solution patterns)
- Database access

6.9.4 Dependencies

This section is dependent upon Inter-task Communication (WBS 1.5.11.1), Legacy Code (WBS 1.5.6.1), Significant Event/Alarm System (WBS 1.5.6.3), EPICS System (WBS 1.5.6.4), ACNET Front-End System (WBS 1.5.6.7), and Control Database (WBS 1.5.6.8).

6.9.5 Schedule

The schedule is undetermined.

6.9.6 Resources/Manpower

The principle resource for this project is programming time according to the following table:

Project	**WBS**	**Related WBS**	**Effort (Person-Weeks)**
Detector support library	1.5.6.10.1	None	8
Silicon vertex tracker		1.1.1	12
Scintillating fiber tracker		1.1.2	12
Preshower		1.1.3-4	12
Calorimeter		1.2	12
Muon		1.3	12
Trigger		1.4	12
Total			80

Other than components of the detector support library, which contains utilities that are shared across one or more detectors, the majority of the manpower for this project is expected to come from the individual detector groups.

6.9.7 Expenditures

6.9.8 Equipment

The equipment costs for this project will come from the individual detector budgets.

6.9.9 Software

There are no anticipated commercial software costs.

7. Accelerator Interface

7.1 Accelerator Console

The author of this section is Laura Paterno. The WBS section is 1.5.?.?.

7.1.1 Introduction

The Accelerator control room provided DØ with information vital to doing luminosity calculations during Run I. This information included xxx. The information was stored on the online host in a database every few minutes.

7.1.2 Requirements

- A connection to the Accelerator ACNET database that is reliable.

7.1.3 Dependencies

This section is dependent on the online database (WBS 1.5.6.10) and the interprocess communications package, ACE (WBS 1.5.6.10).

7.1.4 Schedule

The schedule is undetermined.

7.1.5 Resources/Manpower

This project is not simple and will require a physicist (grad student, post doc, or PhD) with knowledge of the accelerator and the how to calculate luminosity.

Project	**Effort (Person Weeks)**
Total	

7.1.6 Equipment Cost

????? Information Exchange

8. Hardware